VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Filip Navrátil

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Filip Navrátil**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Enwox Systems s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Konzultant bakalářské práce: Filip Siwiec

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016

ENWOX SYSTEMS s.r.o.
Jantarová 3347/11,
702 00 Ostrava - Moravská Ostrava
IČ: 02944049, DIČ: CZ02944049

Rád bych touto cestou chtěl poděkovat firmě ENWOX SYSTEMS s.r.o, za umožnění vykonání bakalářské práce, a dále členům vývojářského týmu a mému odbornému konzultantovi Filipu Siwiecovi, za ochotu a odbornou pomoc při řešení pracovních úkolů. A také vedoucímu mé bakalářské práce Ing. Davidu Ježkovi, za odborné vedení a konzultace.

Abstrakt

Práce popisuje absolvování odborné praxe ve firmě ENWOX SYSTEMS s.r.o. za účelem vytvoření funkčního docházkového systému na míru. Jsou zde uvedené technologie a postupy, které byly použity pro splnění zadaného úkolu. Práce je rozdělená do pěti kapitol. První kapitola popisuje zaměření firmy a mé pracovní zařazení. V druhé kapitole jsou popsány použité technologie pro vývoj IS. Další tři kapitoly se věnují už samotnému zadání úkolu a jeho následného řešení.

Klíčová slova: odborná praxe, docházkový systém, komponentový systém, bakalářská práce, PHP, Nette, MySQL, HTML, CSS, ENWOX SYSTEMS s.r.o, MVC

Abstract

The thesis describes the completion of professional practice in the firm ENWOX SYSTEMS s.r.o. to create a functional system to measure attendance. There are these technologies and practices that have been used to fulfill the task. The work is divided into five chapters. The first chapter describes the focus of the company and my job title. The second chapter describes the technologies used for the development of IS. The next three chapters deal with longer alone task and its subsequent solutions.

Key Words: professional practice, attendance system, component system, bachelor thesis, PHP, Nette, MySQL, HTML, CSS, ENWOX SYSTEMS s.r.o, MVC

Obsah

Seznam použitých zkratek a symbolů

CSRF	–	Cross-site Request Forgery
CSS	–	Cascading Style Sheets
EU	–	Evropská Unie
HTML	–	HyperText Markup Language
HTTP	–	Hypertext Transfer Protocol
IS	–	Informační systém
IT	–	Informační technologie
MVC	–	Model-View-Controller
MVP	–	Model-View-Presenter
MySQL	–	My Structured Query Language
PHP	–	Hypertext Preprocessor
POP3	–	Post Office Protocol
SMTP	–	Simple Mail Transfer Protocol
UML	–	Unified Modeling Language
XHTML	–	Extensible Hypertext Markup Language
XML	–	Extensible Markup Language
XSS	–	Cross-site scripting

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Odbornou bakalářskou práci jsem si vybral z důvodu, abych získal nové praktické zkušenosti v oblasti IT a měl jsem snadnější vstup na trh práce. Další výhodou pro mne byla další možná spolupráce s touto firmou.

Praxi jsem vykonával ve firmě ENWOX SYSTEMS s.r.o.. Do této firmy jsem nastoupil v době, kdy jsem ještě nevěděl, že u této firmy budu mít možnost vykonat bakalářskou práci formou praxe. Nastoupil jsem na pozici IT support a tester IS. Během mého působení v této firmě mi bylo nakonec dovoleno absolvovat bakalářskou praxi na pozici PHP vývojář. Mým úkolem bylo vytvořit IS pro evidenci docházky zaměstnanců skupiny Enwox.

Tento systém by měl pomoct usnadnit evidenci docházky a do budoucna usnadnit firmě vypočet mezd jejich zaměstnanců. Hlavním funkcionalitou systému má být evidence všech zaměstnanců skupiny ENWOX a jejich docházky. Systém má pomoci eliminovat zápis docházky na papír, nebo do tabulek v programu MS Office Excel a usnadnit její zpracování. Dále má pomoci zjednodušit vyřizování požadavků na absence zaměstnanců, a to týkajících se návštěv lékařů, dovolených, homeoffice, apod... Měl by být také nápomocen vedoucím lépe kontrolovat své podřízené, pomocí on-line stavu zaměstnanců. A v neposlední řadě by měl mít také propojení na HW terminály.

V této bakalářské práci se věnuji popisu firmy a mého pracovního zařazení. Dále pak popisu projektu, postupů a technologií, které jsem použil při řešení jednotlivých úkolů.

Projekt je zaměřen pouze na evidenci docházky a to z důvodu, že firma prozatím nepožaduje rozšíření tohoto systému o vypočítávání mezd zaměstnanců.

2 Zaměření firmy a pracovní zařazení studenta

2.1 O firmě

Firma ENWOX SYSTEMS s.r.o. vznikla v roce 2014. Firma se zabývá tvorbou webových prezentací, informačních systémů a internetových aplikací. Avšak hlavním posláním je poskytování IT služeb a vývoj aplikací pro skupinu ENWOX, která se zabývá vývojem a výrobou inteligentní technologie pro stabilizaci a regulaci energetických sítí, smart metering, smart grid a je také dodavatelem elektrické energie a plynu.

Tato společnost patří mezí menší firmy, v jejíž kolektivu se nachází 20 vývojářů, kodérů a grafiků. Má k dispozici vlastní vývojové centrum v Ostravě a udržuje spolupráci s místní Technickou univerzitou. Působnost firmy je nejen na území České republiky, ale její systémy jsou používány i na Slovensku a Polsku. Do budoucna firma předpokládá expanzi se svými systémy do dalších zemí EU [1].

2.2 Pracovní zařazení studenta

Ve firmě jsem byl zařazen do PHP týmu pod vedením mého konzultanta Filipa Siwiece. Primárně jsem byl zaměstnán na pozici IT support a tester IS. Po schválení této bakalářské práce mi bylo umožněno po dohodě s vedoucím oddělení pracovat v týmu, který se věnuje vývoji IS. Na tomto oddělení jsem mohl pracovat vždy až ke konci týdne, po splnění pracovních povinností na mé primární pozici. Proto jsem také praxi vykonával nejen ve firmě ale také i doma. Doma jsem se věnoval především samostudiu, které obnášelo zdokonalování používaných technologií, v hledání způsobů řešení úkolů a také implementaci úkolů, které mi byli během praxe zadány.

3 Použité technologie a nástroje

3.1 Databáze, server

Během mé praxe jsem využíval nakonfigurovaný webserver Apache a MySQL databázi. Ten jsem si opatřil nainstalováním balíku XAMPP [2]. Pro práci s MySQL databází jsem využíval nejrozšířenější prostředí phpMyAdmin. Tato databáze uchovává informace o uživatelích, uživatelských účtech, struktuře oddělení a docházce.

3.1.1 MySQL

MySQL je multiplatformní databáze. Komunikace probíhá pomocí jazyka SQL. Pracoval jsem s verzí 5.6.26.

3.2 PHP

Při vývoj IS jsem pracoval s jazykem PHP a to s jeho nejaktuálnější verzí 5.6.12. Je to nejrozšířenější skriptovací programovací jazyk, který je určen pro programování dynamických internetových stránek a webových aplikací. Při použití PHP jsou scripty prováděny na straně serveru, uživatel vidí až výsledek jejich činností. Dále podporuje mnoho knihoven pro zpracování textu, grafiky, práci se soubory, přístupy k databázím (MySQL, Oracle) a podporuje celé řady internetových protokolů (HTTP, SMTP, POP3, ...) [3] [4].

3.3 Framework Nette

Framework je softwarová struktura, která slouží jako podpora při programování a vývoji softwarových projektů. Může obsahovat podpůrné programy, knihovny a podporu pro návrhové vzory nebo doporučené postupy při vývoji. Pro práci jsem použil framework Nette a pracoval jsem s verzí 2.2.12. Nette je moderní MVC(MVP) framework pro PHP, který výrazně zjednodušeje tvorbu webových aplikací. Tento framework aplikaci rozděluje do komponent tří typů, které se starají o řízení, logiku a výstup. Popis principu MVP je v kapitole 3.8.

Nette framework má několik výhod. Jedna z největších výhod tohoto frameworku je ta, že je vyvíjen českou komunitou (dokumetace v češtině) a jedná se o Open-Source licenci, takže je dostupný zdarma a je možné ho využívat i v komerčních projektech. Další výhody jsou, že eliminuje výskyt bezpečnostních děr a jejich zneužití (útoky XSS, CSRF), disponuje ladícími nástroji, které odhalují chyby aplikace. Využívá různé pluginy a rozšíření, a obsahuje databázové frameworky (NotORM, Dibi). Další výhoda je ta, že poptávka po Nette vývojářích v České republice hodně stoupá.

Bohužel Nette má několik nevýhod. Jednou z nevýhod je jeho nesrovnatelně menší uživatelská základna oproti jiným frameworkům jako je např. Symphony, Zend, apod. [5].

3.4 Front-end

Pro vytvoření vizuální části aplikace jsem použil následující jazyky a knihovnu.

3.4.1 Twitter Bootstrap

Bootstrap je sada nástrojů pro tvorbu webu a webových aplikací. Tato knihovna obsahuje návrhářské šablony založené na HTML a CSS [6].

3.4.2 HTML

HTML je značkovací jazyk pro tvorbu webů. Je to jeden z hlavních jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu [7].

3.4.3 CSS

CSS je jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML [8].

3.5 OBO

OBO je Open-Source projekt vyvíjen firmou ENWOX SYSTEMS s.r.o.. Jedná se o PHP knihovnu, která řeší doménovou logiku. Hlavním cílem je zrychlit a zkvalitnit vývoj. Knihovna se neustále vyvíjí dle aktuálních požadavků.

3.6 FEHU

Fehu je firmou vytvořená platforma, která funguje jako prostředník mezi frameworkem, na kterém se systémy vyvíjí, a koncovými programátory. Základy platformy vznikly na dlouholetých zkušenostech vývojářů a analýze aktuálních požadavků. Hlavním cílem je zrychlit a zkvalitnit vývoj, udělat systémy bezpečnější a snáze modifikovatelné. Tato platforma se snaží neustále pokrývat víc a víc požadovaných funkčností, tak aby se standardizoval vývoj a psáni kódu mezi jednotlivými vývojáři. Řeší různé bezpečnostní otázky, poskytuje mnoho hotových řešení, které vývoj systému podstatně urychlují. Platforma se neustále vyvíjí dle aktuálních požadavků, tak aby se programátoři mohli věnovat konkrétním problémům ve svých projektech, urychlil se vývoj a nebylo potřeba vývoj zpomalovat řešením nízkoúrovňových věcí, které pravě Fehu řeší za ně.

3.7 GIT

Git je verzovací nástroj. Pro práci s GITem jsem využíval klienta SmartGit, který mi umožnoval rychlou a snadou práci [9].

3.8 MVP

V Nette frameworku se používá název MVP(model-view-presenter) místo MVC(model-view-controller), ale princip je stejný. MVP je softwarová architektura, která rozděluje aplikaci do tří nezávislých vrstev – prezentační, aplikační a datovou. Díky nezávislosti těchto vrstev, úprava jedné nezpůsobí zásadní problém v dalších vrstvách. Avšak v ojedinělých případech mohou změny jedné vrstvy ovlivnit ostatní [10].

- Model – obsahuje doménovou logiku aplikace. Nachází se zde metody pro práci s databází nebo různé výpočty. Každá entita má svůj vlastní model.
- View – prezenční vrstva, převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli. Obsahuje Latte šablony s HTML kódem.
- Presenter – řídíci prvek, který má přístup k View a Modelu a řídí jejich vzájemnou komunikaci.

4 Zadání

Požadavkem firmy ENWOX SYSTEMS s.r.o. bylo vytvořit docházkový systém, vytvořený pro tuto firmu na míru. Systém má být naimplementován pomocí webových technologií a bude postaven na Nette frameworku, který dodržuje MVP architekturu.

Na začátku vývoje jsem dostal od firmy pouze předběžné zadání, co by IS měl zvládat a evidovat. Během vývoje se zadání postupně rozšiřovalo a měnilo, vzhledem k narůstajícím požadavkům firmy.

4.1 Funkční požadavky

- Bezpečné přihlášení uživatele do systému.
- Logování významných událostí systému.
- Správa uživatelských účtů.
- Správa struktury organizace.
- Rychlé vložení docházky (dostupná i z mobilního zařízení).
- Možnost vložení docházky (příchod/odchod, oběd, dovolené, návštěva lékaře, či jiné přerušení docházky, atd.).
- Emailové notifikace.
- Přehledy odpracovaných hodin (denní, týdenní, měsíční).
- Statistiky jednotlivých oddělení a zaměstnanců.
- Exporty sestav do různých formátů.
- Exporty pro účetní systémy.
- API pro vložení docházky (příchod/odchod) z HW terminálu.

5 Postup a řešení jednotlivých úkolů

Během vývoje byly úkoly rozděleny dle priority do několika skupin. Prvotně musely být vyřešeny uživatelské účty, přihlašování do systému a struktura firmy. Po naimplementování tohoto základu systému, mohla být řešena otázka evidování docházky. Vývoj probíhal v iteracích. Pro každou iteraci byl zadán úkol, po kterém následovalo plánování a návrh jak, by měl být úkol řešen. Poté následovala implementace a nakonec proběhlo testování.

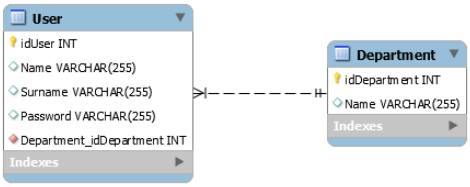
Během praxe, jsem se účastnil porad, na kterých byly řešeny způsoby řešení jednotlivých úkolů, které byly firmou zadány. Na poradách byl prvotně přednesen můj návrh řešení úkolu, který jsem vypracoval v rámci samostudia. V průběhu se buďto řešení upravilo nebo bylo rovnou schváleno mým konzultantem a vedoucím vývojového týmu. Až poté mohla započít samotná implementace daného úkolu.

5.1 Obecný postup

Při řešení konkrétních úkolů byla práce rozdělena do několika kroků, které na sebe navzájem navazovaly. V této části bych chtěl nastínit obecný postup pro vypracování jednoho zadaného úkolu.

5.1.1 Návrh databáze

Při zpracování některých úkolů musela být navrhnuta databáze. Návrhy databáze byly prováděny v programu MySQL WorkBrench [11]. V návrhu byly vždy znázorněny tabulky, sloupce a vazby mezi tabulkami. Databáze byla postupným vývojem rozšiřována a měnila se vzhledem k požadavkům firmy. Jelikož se tyto požadavky neustále měnily, přibývaly nebo ubývaly tabulky a sloupce a měnily se vazby, nebylo hned zpočátku možné vytvořit jeden prvotní návrh, který by obsahoval vše. Na obrázku 1 jsou znázorněny dvě tabulky a vazba 1:N.



Obrázek 1: Návrh tabulek

5.1.2 Modely

Po návrhu databáze byl proveden další krok, a to, psaní modelů. Jeden model odpovídal vždy jedné tabulce v databázi. Každý model byl rozdělen do tří tříd.

- V první části modelu jsou naimplementovány základní CRUD metody a další metody, které byly potřebné pro práci s DB.
- Druhá část modelu obsahovala pomocné funkce pro obsluhu dané entity.
- Třetí část modelu obsahovala property, které odpovídaly sloupcům dané tabulky v databázi, nacházely se zde i pomocné property. Nad každou property jsem musel mít definované OBO anotace. Tyto anotace sloužili např. k nastavení typu vazby, datového typu property a různých nastavení.

5.1.3 Komponenty

V tomto projektu byl použit komponentový systém. Což znamená, že každá stránka se skládá z mnoha komponent. Tyto komponenty o sobě navzájem nemusejí vědět, ale pokud chtějí, mohou spolu komunikovat. Každá komponenta má vlastní view a controller, který přistupuje k modelům. Komponentový systém ve webové vrstvě přináší možnost elegantní znovupoužitelnosti komponent.

View

View je realizován pomocí šablonovacího systému Latte. Využívání šablon má několik hlavních cílů, a to např. zpřehlednění HTML a PHP kódu nebo jednoduché oddělení kódu aplikace od prezentace výstupů, tak aby změna v jedné části nevedla k chybám v druhé. Další výhodou je automatické escapování výstupů. Latte také své soubory cachuje, dělá to pouze při prvním načtení, tudíž se pak veškeré renderování posléze urychlí. Nevýhodou avšak je, že pro použití musíte znát syntaxi šablonového systému, která je sice podobná PHP, ale není úplně stejná [12]. Já osobně jsem se musel tuto syntaxi naučit a bylo to poměrně jednoduché. V výpisu kódu 1 a 2 je porovnání, jak vypadá výpis uživatelů s použitím šablonovacího systému Latte a bez.

Realizace View byla jednoduchá. Vždy jsem pomocí syntaxe Latte a jazyku HTML vydefinoval vzhled stánky. Latte poté vygeneroval výsledný HTML kód. Prohlížeč pak pomocí tohoto kódu, CSS stylů a JavaScriptu vykreslil stránku. Pomocí nastavených Latte proměnných, předal Controller data do View. Ve výpisu kódu 1 je uvedena ukázka syntaxe Latte (jednoduchý foreach cyklus), která vypisuje přehled uživatelů a obsah property *name* a *surname*. Data do proměnné *$users* předal contoller z modelu pomocí metody, která je znázorněna ve výpisu kódu 3.

```
{define content}
 <h1>Uzivatele</h1>
  <table class="table table-bordered table-striped table-hover">
    <thead>
      <tr>
        <th>Jmeno</th>
        <th>Prijmeni</th>
      </tr>
    </thead>
    <tbody>
      {foreach $users as $user}
        <tr>
          <td>{$user->name}</td>
          <td>{$user->surname}</td>
        </tr>
      {/foreach}
    </tbody>
  </table>
{/define}
```

Výpis 1: Ukázka syntaxe Latte

V následujícím výpisu kódu 2 je znázorněn výpis uživatelů bez šablonovacího systému Latte. Na první pohled je zřejmé, že jednou z výhod šablonovacího systému je zpřehlednění kódu.

```
<h1>Uzivatele</h1>
<table class="table table-bordered table-striped table-hover">
  <thead>
    <tr>
      <th>Jmeno</th>
      <th>Prijmeni</th>
    </tr>
  </thead>
  <tbody>
    <?php foreach ($users as $user) : ?>
    <tr>
      <td><?= $user['name'] ?></td>
      <td><?= $user['surname'] ?></td>
    </tr>
  <?php endforeach ?>
  </tbody>
</table>
```

Výpis 2: Ukázka bez použití šablonovacího systému Latte

Controller

V Controlleru jsou naimplementovány metody potřebné pro spolupráci Modelu a View. Controller dokáže zpracovat data, která buďto přichází od uživatele, nebo jsou vrácena modelem. Data vrácena modelem předává do View pomocí proměných, které jsou definované v Latte šablonách.

Ve výpisu kódu 3 je znázorněna metoda prepareDataForRender(), která předává získané data z Modelu do View. V ukázce vidíme, jak controller předává data z modelu (model pomocí metody getAllUsers() vytáhne objekty z DB), do proměné *users*. Poté je zapotřebí nadefinovat proměnou i v Latte šabloně, která nám data, poskytnuté modelem vypíše. Tato proměná je pojmenovaná *$users* a můžeme ji vidět ve výpisu kódu 1.

```
public function prepareDataForRender() {
  $this->template->users = \Models\Users::getAllUsers();
}
```

Výpis 3: Ukázka předání dat z Modelu do View

5.1.4 Testování

Testování je důležité při vývoji jakéhokoliv softwaru. Cílem testování je objevit a opravit co nejvíce chyb. Avšak toto nezaručí bezchybnost aplikace, jelikož není reálné nasimulovat všechny možné vstupy, výstupy a různé situace. Přesto lze testováním objevit většinu chyb, na které může běžný uživatel narazit.

Testování probíhalo během celého vývoje systému. Bylo rozděleno do dvou fází.

V první fázi probíhalo uživatelské (manuální) testování. Zde se musely ověřit funkční požadavky aplikace. To znamená, že bylo nutné otestovat, zda naimplementovaná funkcionalita odpovídá funkčním požadavkům z pohledu uživatele systému. Ověřovala se také správnost činností dílčích částí, a také se ověřovali vstupy a výstupy.

Tester dostal testovací scénář, ve kterém byl uveden popis nové funkcionality. V tomto popisu bylo uvedeno, jak by se měla tato funkcionalita správně chovat. Byly zde uvedeny správné vstupy, výstupy nebo podmínky správného chování. Pokud funcionalita neodpovídala požadavkům bylo nutné tyto chyby opravit a poté znova otestovat.

Firma závádí testování pomocí Selenium testů. Toto zautomatizované testování postupně nahrazuje manuální a stává se standardem. Proto během vývoje IS jsem také přešel na toto testování pomocí Selenium testů. Automatické testování probíhá na principu psaní jednotlivých kroků scénaře, kdy po spuštění testu se tyto kroky provedou a aplikaci otestuje obdobně jako uživatel. Toto je jedna z hlavních výhod.

Nevýhodou tohoto testování je zdlouhavé psaní scénářů a naučení se, jak správně tyto testy psát, aby měli co největší efektivitu. Je proto lepší psát více menších a jednoduchých scénářů než jeden velký a složitý.

V druhé fázi proběhlo testování samotného kódu. Nejprve proběhla statická analýza kódu. Tyto testy měly za úkol analyzovat kód aplikace a upozornit na chyby v coding standardu. Coding standard je velmi důležitý, proto aby se udržela čitelnost kódu. Pro tyto testy jsem používal nástroj CodeSniffer.

5.1.5 Grafika

Jelikož grafika není mým oborem, tak celou dobu jsem spolupracoval s firemním grafikem. Ze začátku jsem grafikovi pouze představil systém, a nechal jej, ať sám navrhne jeho vzhled. Grafik navrhl hlavní layout systému a nadefinoval barvy, které jsem při vytváření vzhledu měl použít. Posléze jsem s grafikem spolupracoval častěji, představoval jsem mu jednotlivé stránky systému. On pak pro tyto stránky navrhl vzhled včetně barev. Já jsem poté podle těchto návrhů nastyloval jednotlivé stránky. Na konci byla má práce grafikem zhodnocena a byly opraveny mé chyby.

K realizaci struktury stránky jsem použil jazyk HTML. Dále jsem použil dostupnou knihovnu Bootstrap, která byla částečně modifikována pomocí CSS stylů.

5.2 Postup řešení jednotlivých úkolů

5.2.1 Uživatelský účet

Pvním krokem bylo vyřešit vytvoření nového uživatele a bezpečné přihlášení do systému. K vytvoření nového uživatele bylo zapotřebí zadat jméno, příjmení, login a heslo, které muselo obsahovat minimálně 8 znaků, z toho jedno velké písmeno a číslici. Bezpečnost přihlášení řešila platforma Fehu, mnou byly pouze nadefinovány podmínky přihlášení.

Druhým krokem bylo vytvoření samotného uživatelského účtu. Zde bylo potřeba vědět jakým způsobem a jaké informace se budou evidovat. Na základě tohoto proběhla analýza, ze které se určila struktura uživatelského účtu. V tomto účtu se zaprvé evidovaly základní informace o samotném uživateli (jméno, příjmení, datum narození, email, telefon, adresa bydliště, oddělení, pracovní zařazení). Dále pak uživatelské kontakty. U každého uživatele může být evidováno několik kontaktů. Každý kontakt se skládá z minimálně jedné adresy, emailu a telefonu. Toto komplexní rozvržení je zapotřebí z důvodu evidování jak pracovních tak i osobních kontaktů.

Dalším rozšířením účtu bylo např. evidování zástupců uživatele nebo elektronického uchovávání důležitých dokumentů (smlouvy, propustky od lékaře, neschopenky, atd...).

5.2.2 Stromová struktura

Tuto strukturu bylo v tomto projektu zapotřebí použít a to z důvodu, aby byla zaevidována hierarchická struktura oddělení ve firmě. Později se pomocí této struktury evidovala i hierarchická struktura zaměstnanců v daném oddělení.

První problém, který se vyskytl, byl problém jak navrhnout databázi, tak aby bylo jednoduché ukládat strom, uzly a pozici uzlu ve stromě. I tuto problematiku jsem si musel podrobněji nastudovat v rámci samostudia [13]. Po jejím nastudování, jsem strukturu databáze navrhl poměrně jednoduše, kdy strom zná pouze svůj kořen. Dále pak v této stromové struktuře každý uzel ve stromě má informaci o svém rodiči a všech svých potomcích, a také o obsahu. V první fázi mohl obsah evidovat pouze entitu, která reprezentovala samotné oddělení. Následně měla firma požadavek, aby byla evidována i hierarchická struktura zaměstnanců, muselo dojít k úpravě stromu. Proto zde došlo ke druhé fázi, kdy byl obsah implementován univerzálně, a to tak, že do obsahu může být přidána jakákoliv entita.

5.2.3 Evidence docházky

Jakmile byla vyřešena struktura firmy a oddělení, evidence zaměstnanců a dalších informací, mohlo se přejít k samotné evidenci docházky. Bylo nelehké navrhnout, jakým způsobem se budou do databáze ukládat data. A to proto, že se muselo do návrhu databáze zahrnout několik faktů.

Prvním faktem bylo vyřešit jak evidovat jednotlivé časové úseky. Po poradě s nadřízeným a mým konzultantem jsme došli k závěru, že do databáze bude zapisován začátek a konec jednotlivých časových intervalů.

Druhým faktem, který musel být do návrhu databáze zahrnut, bylo, aby systém mohl v reálném čase počítat statistiky a přehledy za jednotlivé časové období (denní, týdenní, měsíční). Do databáze musely být tedy přidány tabulky, které umožňovaly uchovat potřebné data. Tyto tabulky byly v podstatě cache pro možnost rychlého výběru potřebných dat. Zahrnutí těchto časových úseků do databáze mělo výhodu v tom, že každý z těchto intervalů měl konstantní délku (denní, týdenní, měsíční).

Samozřejmě toto přineslo i nevýhodu a to v tom, že se musel vytvořit algoritmus, který by tyto časové úseky vypočítal.

Mnou navrhnutý algoritmus pracuje na principu vyhledávání začátku a konce daného časového úseku, např. k začátku časového úseku „příchod do práce“ algoritmus vyhledá konec časového úseku „odchod z práce“.

5.2.4 Emailové notifikace

Dalším požadavkem firmy bylo naimplementovat emailové notifikace na různé akce a stavy v systému. Tyto emailové notifikace sloužily k upozornění uživatele (vedoucího, zaměstnance), např. k dopsání a uzavření docházky, k upozornění o nepřítomnosti zaměstnance nebo k ohlášení absence.

K vyřešení této problematiky, jsem musel správně v konfiguračním souboru nastavit SMTP server. Pro odchytávání emailu jsem použil program SMTP4dev [14]. Většinu problematiky řešila platforma FEHU, kdy jsem pouze dědil ze třídy MAILER a poté jen psal jednoduché metody, kde jsem uvedl příjemce, předmět a obsah zprávy. Ve výpisu kódu 4 je znázorněná jednoduchá metoda sendMail(), která odešle email danému uživateli.

```
public function sendMail() {
  $this->sendSimpleMail($user->email, "Oznameni", "Dobry den, za den ".$date."
      nemate zapsanou dochazku");
}
```

Výpis 4: Metoda pro odeslání emailu

5.2.5 Export do xlsx

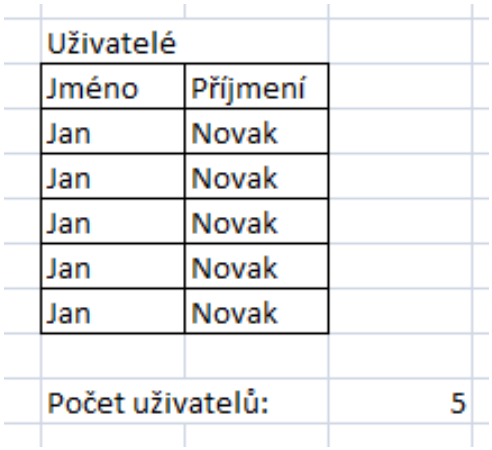
V IS bylo zapotřebí naimplementovat export různých dat do formátu xlsx. Tímto způsobem se nejčastěji exportoval měsíční přehled docházky zaměstnance. Exportování dat do xlx bylo velmi zajímavé, jelikož tato knihovna obsahuje spoustu funkcí pro práci se souborem. Bylo zde možné naformátovat např. jednotlivé buňky excelu, nadefinovat šířku / výšku buněk, ohraničení, sloučení a nebo přesně nadefinovat do jaké buňky se má jaká hodnota vepsat. Abych tento úkol mohl splnit musel jsem si nastudovat knihovnu PHPExcel [15].

Ve výpisu kódu 5 můžeme vidět vytvoření nového souboru, jednoho listu a naformátování a vyplnění jednotlivých buněk. Výsledek tohoto kódu můžeme vidět na obrázku2.

```
public function Export(){
  $excel = new \PHPExcel();
  $list  = $excel->getActiveSheet();
  $list ->setTitle("Export");
  $list ->mergeCells("B3:C3")->setCellValue("B3","Uzivatele", false);
  $list ->setCellValue("B4","Jmeno")->getStyle()->getFont()->setBold(true);
  $list ->setCellValue("C4","Prijmeni")->getStyle()->getFont()->setBold(true);
  for($i=5; $i<10; $i++){
    $list ->setCellValue("B".$i,"Jan");
    $list ->setCellValue("C".$i,"Novak");
  }
  $list ->getStyle('B4:C9')->getBorders()->getAllBorders()->setBorderStyle(\
      PHPExcel_Style_Border::BORDER_THIN);
  $list ->mergeCells("B11:C11")->setCellValue("B11","Pocet uzivatelu:", false);
  $list ->setCellValue("D11","=COUNT(B5:B9)");

  $objWriter = new \PHPExcel_Writer_Excel2007($excel);
  $objWriter->save('export.xlsx');
}
```

Výpis 5: Ukázka exportu do xlsx



Uživatelé	
Jméno	Příjmení
Jan	Novak
Jan	Novak
Jan	Novak
Jan	Novak
Jan	Novak
Počet uživatelů:	5

Obrázek 2: Výsledek exportu do xlsx

5.2.6 ACL

ACL se odvíjelo od předem určených uživatelských rolí, které budou se systémem pracovat. ACL bylo řešeno formou vlastnictví. Ke každé komponentě bylo nadefinoných N módů a na základě podmínek u přihlášeného uživatele se komponenta přepla do patřičných módů, které se pak starají co přihlášený uživatel může / nemůže.

Ve výpisu kódu 6 můžeme vidět zdrojový kód Latte šablony. V šabloně kontroluji zda přihlášený uživatel má v módech povolené různé části komponenty. Módy této komponenty jsou ve výpisu kódu 7.

```
{if $control->isAllowed('addUser')}
    <a n:alink="addUser!">{"Novy uzivatel"}</a>
{/if}
{snippet datagrid}
  {if $control->isAllowed('filter')}
    {control filter }
  {/if}
   <table n:if="count($users)">
       <tr>
           <th>{"Jmeno"}</th>
           <th>{"Prijmeni"}</th>
       </tr>
   </table>
{/snippet}
```

Výpis 6: Kontrola módů v šabloně

Ve výpisu kódu 7 jsem nadefinoval dva různé módy komponenty. Tyto módy se na základě dané podmínky přiřadí přihlášenému uživateli.

```
public static function getModesStructure() {
  return [
    "modY" => [
      " filter ",
    ],
    "modX" => [
      "addUser",
      " filter ",
    ],
  ];
}

public function attached($presenter) {
  parent::attached($presenter);
  if ($this->presenter->user->identity->getObject()->role === 1){
      $this->getModesStructure("modY");
   } elseif ($this->presenter->user->identity->getObject()->role === 2){
          $this->getModesStructure("modX");
   }
}
```

Výpis 7: Definování módů

6 Použité a chybějící znalosti

6.1 Využité znalosti

V průběhu praxe jsem využil mnoho zkušeností, které jsem získal při studiu. První znalost jsem využil při vytváření návrhů databáze. Tuto znalost jsem získal z předmětů Úvod do databázových systémů, Databázové a Informační systémy a Vývoj informačních systémů. Z těchto předmětů jsem také měl dostatečné základní znalosti MVC architektury. Další znalosti, a to vytváření UML diagramů, jsem čerpal z předmětu Úvod do Softwarového inženýrství.

Z předmětů Základy programování, Programovací jazyky 1 a 2 jsme získal dostanečené základy syntaxe programovacích jazyků, proto pro mne bylo jednoduché se naučit syntaxy nového programovacího jazyka PHP.

Jelikož IS pro evidenci docházky je webovou aplikací využil jsem také zkušenosti HTML, CSS a javascriptu, které jsem získal v předmětu Vývoj internetových aplikací, kde jsem se s touto problematikou seznámil. V tomto předmětu jsem také získal znalosti ohledně Bootstrap. Samozřejmě bylo potřeba mé znalosti rozšířit, proto jsem využíval online dokumentace [16] [17] [18] [19].

6.2 Chybějící znalosti

Ze začátku jsem musel doplnit znalost jazyku PHP, frameworku Nette a šablonovacího systému Latte. Tuto problematiku jsem nastudoval za pomoci různých webů např. z online tutoriálů nebo odborné dokumentace [20] [21] [22] . Při doplnění chybějících znalostí mi pomohl můj konzultant pan Filip Siwiec, který mi doporučoval vhodnou dokumentaci k samostudiu i programy, které jsem během mé praxe využíval. Pomohl mi především při práci s verzovacím nástrojem GIT, kdy mne naučil postupy správného pushování a mergovaní. A také mi pomohl doplnit znalosti praktického řešení ACL. Dalé jsem si doplnil znalosti ohledně knihovny PHPExcel, k doplnění znalostí jsem využil dokumentace [15].

7 Závěr

Za dobu mé praxe ve firmě ENWOX SYSTEMS s.r.o. jsem vytvořil systém, který má na starost evidenci docházky zaměstnanců skupiny ENWOX. Během vývoje systému jsem si prohroubil své znalosti. Díky tomuto jsem získal cené praktické zkušenosti, které budou potřeba pro moji budoucí kariéru.

Samotný vývoj IS byl náročný, avšak díky dobrému pracovnímu kolektivu firmy jsem dokázal tuto náročnost překonat. Proto bych zde na závěr chtěl poděkovat Filipu Siwiecovi a celému vývojářskému týmu za jejich cené rady a nápomocnost při vývoji.

Samotný IS funguje na principu zapisování příchodů a odchodů zaměstnanců. Avšak není nasazen v reálných podmínkách, a to z toho důvodu že u něj neproběhli všechny potřebné testy. V této chvíli je avšak položen dobrý základ pro postupné rozšiřování IS, a tím pádem by mohl později nahradit lidský faktor např. při vypočítávání mezd, evidenci zaměstnanců atd... .

Svou práci hodnotím pozitivně, jelikož vytvořením tohoto systému jsem položil základ pro zjednodušení administrativní práce.

Literatura

[1] Základní údaje o firmě ENWOX SYSTEMS s.r.o.. *ENWOX SYSTEMS s.r.o.* [online]. ©2016 [cit. 2016-04-24]. Dostupné z: `https://www.enwoxsystems.cz/o-nas/`

[2] XAMPP. *XAMPP Installers and Downloads for Apache Friends* [online]. ©2016 [cit. 2016-04-24]. Dostupné z: `https://www.apachefriends.org/index.html`

[3] PHP. *PHP: Hypertext Preprocessor* [online]. ©2001-2016 [cit. 2016-04-24]. Dostupné z: `http://php.net/`

[4] PHP - Wikipedia. *Wikipedia, the free encyclopedia* [online]. 23.4.2016 [cit. 2016-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/PHP`

[5] Nette. *Nette framework* [online]. ©2008-2016 [cit. 2016-04-24]. Dostupné z: `https://nette.org/`

[6] Bootstrap. *Wikipedia, the free encyclopedia* [online]. 19.4.2016 [cit. 2016-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)`

[7] HTML. *Wikipedia, the free encyclopedia* [online]. 22.4.2016 [cit. 2016-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/HTML`

[8] Cascading Style Sheets. *Wikipedia, the free encyclopedia* [online]. 22.4.2016 [cit. 2016-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/Cascading_Style_Sheets`

[9] Git Client SmartGit. *Syntevo* [online]. [cit. 2016-04-24]. Dostupné z: `http://www.syntevo.com/smartgit/`

[10] Model–view–controller. *Wikipedia, the free encyclopedia* [online]. 22.4.2016 [cit. 2016-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/Model-view-controller`

[11] MySQL::MySQL Workbench. *MySQL* [online]. ©2016 [cit. 2016-04-24]. Dostupné z: `http://www.mysql.com/products/workbench/?page_id=35`

[12] Nette Framework: Chytré šablony. *Zdroják.cz* [online]. 7.4.2009 [cit. 2016-04-24]. Dostupné z: `https://www.zdrojak.cz/clanky/nette-framework-chytre-sablony/`

[13] Storing Hierarchical Data in a Database. *SitePoint* [online]. ©2000-2016 [cit. 2016-04-24]. Dostupné z: `http://www.sitepoint.com/hierarchical-data-database/`

[14] smtp4dev. *CodePlex* [online]. ©2006-2016 [cit. 2016-04-24]. Dostupné z: https://smtp4dev.codeplex.com/

[15] GitHub - PHPOffice/PHPExcel. *GitHub* [online]. ©2016 [cit. 2016-04-24]. Dostupné z: https://github.com/PHPOffice/PHPExcel

[16] HTML Tutorial. *W3Schools Online Web Tutorials* [online]. ©1999-2016 [cit. 2016-04-24]. Dostupné z: http://www.w3schools.com/html/default.asp

[17] CSS Tutorial. *W3Schools Online Web Tutorials* [online]. ©1999-2016 [cit. 2016-04-24]. Dostupné z: http://www.w3schools.com/css/default.asp

[18] JavaScript Tutorial. *W3Schools Online Web Tutorials* [online]. ©1999-2016 [cit. 2016-04-24]. Dostupné z: http://www.w3schools.com/js/default.asp

[19] Bootstrap Tutorial. *W3Schools Online Web Tutorials* [online]. ©1999-2016 [cit. 2016-04-24]. Dostupné z: http://www.w3schools.com/bootstrap/default.asp

[20] PHP 5 Tutorial. *W3Schools Online Web Tutorials* [online]. ©1999-2016 [cit. 2016-04-24]. Dostupné z: http://www.w3schools.com/php/default.asp

[21] PHP. *ITnetwork* [online]. ©2016 [cit. 2016-04-24]. Dostupné z: http://www.itnetwork.cz/php

[22] Výchozí Latte makra. *Nette framework* [online]. ©2008-2016 [cit. 2016-04-24]. Dostupné z: https://latte.nette.org/cs/macros

[23] CHACON, Scott. *Pro Git* [online]. New York: Distributed to the book trade worldwide by Springer-Verlag, c2009 [cit. 2016-04-24]. Expert's voice in software development. ISBN 978-143-0218-340. Dostupné z: https://git-scm.com/book/en/v1